Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications

Sudip Misra, *Senior Member, IEEE,* and Niloy Saha, *Student Member, IEEE*

***Abstract*—In this paper, we consider the problem of task offloading in a software-defined access network where IoT devices are connected to fog computing nodes by multi-hop IoT access-points (APs). The proposed scheme considers the following aspects in fog-computing based IoT architecture — a) *optimal* decision on local or remote task computation, b) *optimal* fog node selection, and c) *optimal* path selection for offloading. Accordingly, we formulate the multi-hop task offloading problem as an integer linear program (ILP). Since the feasible set is non-convex, we propose a greedy-heuristic based approach to efficiently solve the problem. The greedy solution takes into account delay, energy consumption, multi-hop paths, and dynamic network conditions such as link utilization and SDN rule-capacity. Experimental results show that the proposed scheme is capable of reducing the average delay and energy consumption by 12% and 21%, respectively, compared to the state-of-the-art.**

***Index Terms*—Software-Defined Networking, Internet of Things, Quality-of-Service, Fog computing, Task Offloading**

I. Introduction

THE rapid growth of Internet of Things (IoT) technologies has led to the emergence of a variety of latency-critical IoT applications such as smart healthcare, vehicular and industrial automation, and augmented reality [1]. These applications demand substantial computation resources for real-time processing which leads to high energy consumption on resource constrained[1] IoT devices. To address this issue, task offloading using edge computing has emerged as a promising solution [2], [3]. The edge computing paradigm proposes the deployment of resource-rich entities at the network edge, in order to execute tasks offloaded from the resource-constrained IoT devices. Moreover, task offloading using edge computing offers low-latency and flexible computation to IoT devices [4].

In this work, we consider task offloading in the context of fog computing [5], which proposes deployment of compute, storage and networking resources anywhere in the device-to-cloud continuum, while providing abstractions to the underlying communication technology which is useful to address the heterogeneity of IoT.

A. Motivation

Task offloading schemes in existing literature have mainly focused on computation power and energy consumption, without considering the network load on the path from device to edge server. Since IoT protocols such as MQTT-SN [6] and CoAP [7] use application layer retransmissions to ensure reliable delivery, packet-loss due to network congestion will lead to increased energy consumption at the IoT devices. Therefore, the dynamic load on the network must be taken into account while making task offloading decisions. This can be achieved by adopting a software-defined network (SDN) architecture for IoT [8], [9], which offers logically centralized control, abstracted global view of the network conditions, and flexible rule-based forwarding. The network management abstractions provided by SDN allows the SDN controller to collect network information from heterogeneous wireless devices, across different wireless technologies [10], [11]. Therefore, using the abstracted global view of the network, the SDN controller is capable of taking optimal task offloading decisions. Moreover, SDN also offers benefits in terms of network flexibility and simplification of network management, for the orchestration and management of fog-based IoT infrastructures [12], [13]. This makes SDN particularly attractive to address the problem of task offloading in a dynamic IoT scenario.

Therefore, leveraging the concept of SDN, we present a dynamic task offloading scheme for IoT applications in the presence of fog devices. Different from existing literature, we consider a scenario where the offloaded tasks have to traverse a multi-hop path to the fog devices. This is due to the fact that in practice, due to limitations of CAPEX and OPEX, the number of fog devices will be less than the number of IoT access points in the network. Although SDN offers various advantages for task offloading, the limited rule-capacity of SDN switches adversely affects the performance of task offloading schemes. Therefore, in contrast to existing literature on SDN-based task offloading, the proposed solution takes into account the flow-rule utilization to mitigate the effects of limited rule-capacity of SDN switches.

B. Contributions

In this work, we present a dynamic task offloading scheme in software-defined access-networks (SDANs) where IoT devices are connected to fog computing nodes by multi-hop IoT access-points (APs). The SDN controller can collect network information using southbound APIs, and takes optimal task offloading decisions according to its global view of the network. In particular, in a fog-computing based IoT scenario, the proposed scheme considers the following aspects — a) *optimal* decision on local or remote task computation, b) *optimal* fog node selection, and c) *optimal* path selection for offloading. Accordingly, we formulate the dynamic task offloading problem as an integer linear program (ILP) and propose a greedy-heuristic based approach to solve it efficiently. Extensive simulation results are presented to show the effectiveness of the proposed scheme. In summary, the main contributions of this work are as follows:

[1] in terms of energy, computation capability, and storage

- We present a task offloading scheme in software-defined access-networks for IoT applications in the presence of fog devices. We consider a scenario where the number of fog devices is less than the number of APs. The problem is challenging because of the existence of multi-hop paths from the devices to the fog nodes.
- We formulate the multi-hop dynamic task offloading problem as a non-linear optimization problem and then transform it into an equivalent integer linear program (ILP). It takes into account the energy constraints of IoT devices and dynamic network conditions such as link utilization and SDN flow-rule utilization.
- Since the feasible set of the ILP is non-convex, we propose a greedy-heuristic based approach to efficiently solve the problem.
- We evaluate the proposed scheme using the POX SDN controller and the Mininet network emulator. Experimental results show that the proposed scheme is capable of reducing the average delay and energy consumption by 12% and 21%, respectively, compared to the state-of-the-art.

The remainder of the paper is structured as follows. In Section II, we analyze the relevant state-of-the-art. In Section III, we present the system model, including architecture, problem formulation, and greedy-heuristic based solution. Section IV presents the performance evaluation of the proposed scheme. Finally, we conclude the paper in Section V and present directions for future work.

II. Related Work

A. Software-defined Fog Architecture for IoT

In the recent past, SDN has received a lot of attention from industry and academia for managing and orchestrating edge computing and IoT architectures [8], [9]. Sood *et al.* [8] discussed the recent efforts to integrate SDN and IoT and highlighted the advantages of such integration on information acquisition, analysis and decision making in IoT. Gupta *et al.* [14] proposed a software-defined fog middleware which provides abstractions to the heterogeneous fog infrastructure and enables orchestration of fog services while considering end-to-end QoS requirements. Hakiri *et al.* [15] proposed a software-defined wireless fog architecture, in order to minimize delay and provide load-balancing among fog devices. In the proposed scheme the SDN controller is used to collect signal-to-noise ratio (SNR) values from the wireless clients, in order to facilitate traffic engineering among the wireless fog devices. Tomovic *et al.* [13] proposed a software-defined fog architecture consisting of geo-distributed fog-nodes in order to improve the overall performance of an IoT network. The authors highlighted the advantages of the SDN-fog interplay in terms network flexibility and scalability.

Therefore, due to the advantages of SDN in terms of orchestrating and managing fog-based architectures, we adopt a software-defined fog-based architecture for IoT.

B. Task Offloading in Fog

Recent works [16]–[21] also addressed task offloading in fog. Chang *et al.* [16] proposed an energy-efficient task offloading scheme for fog computing. The authors presented a queuing analysis of the task queues at both the mobile devices and edge server. Chiti *et al.* [17] proposed a task offloading scheme for fog computing in IoT, based on matching theory. The authors proposed a distributed algorithm in which each mobile device selects the most suitable fog node, based on transmission, waiting, and computation times. However, the authors did not consider the energy constraints of the devices and the dynamic load on the network. Shah-Mansouri and Wong [18] proposed a game-theoretic approach for task offloading in hierarchical fog-cloud systems. The authors focused on achieving *near optimal* task offloading decisions in the presence of selfish IoT users interested in maximizing their own quality of experience (QoE). Yousefpour *et al.* [19] studied task offloading in general IoT-fog-cloud architecture. The proposed scheme considered inter-fog collaboration and load-sharing in order to reduce the overall service delay.

However, the existing schemes did not take into consideration the dynamic network conditions present in an IoT scenario.

C. Software-defined Edge Computing and Task Offloading

Huang *et al.* [22] proposed a MEC framework for SDN-based LTE networks. The authors proposed the *radio API* abstraction in order to extract parameters such as topology, band, and signal strength from the RAN using the SDN controller. The radio API also enables the SDN controller to modify the underlying network state using the statistics and network information gathered from the RAN. Cui *et al.* [23] proposed a software-defined cooperative task offloading scheme for mobile cloudlets using device-to-device (D2D) communication. In the proposed scheme, the SDN controller is deployed at the LTE gateway in order to gain information about the devices and obtain a global view of network states. Task offload to other devices or edge servers take place according to the instructions of the centralized SDN controller. Chen and Hao [24] presented a task offloading scheme for MEC in an ultra-dense software-defined network. In the proposed scheme, the SDN controller is deployed at the macro-cell base-station in order to obtain global information about devices, base-stations, edge servers and tasks. Zhao *et al.* [25] proposed a SDN-based scheme for optimal cloudlet placement in order to reduce access-delay in IoT. Huang *et al.* [27] proposed an SDN-based framework for V2V offloading in VANETs using the MEC concept. In the proposed work, SDN-enabled road-side units (RSUs) were used to collect contextual information and for centralized management and control of offloading strategy. The existing schemes [22]–[25] highlight the advantages of SDN in a task offloading scenario, in terms of — real-time gathering of statistics, modification of network state, and global view of the network. However, our work differs from the existing literature in the following aspects — a) the existing schemes considered task offloading where the offload target is single-hop away from the devices, while in this work, we consider a more general fog computing framework where the fog nodes may be located one or more hops away from the devices, and b) we take into account the

dynamic network conditions such as link-utilization and SDN rule-capacity, which were not considered.

Synthesis: In Table I, we summarize the existing literature on task offloading. Detailed analysis of the state-of-the art reveals that there exists a research lacuna on software-defined task offloading schemes, while taking into account multi-hop paths, and dynamic network conditions such as link utilization and SDN rule-capacity, as well as delay and energy consumption. Therefore, we propose a dynamic task offloading scheme in SDN, named DETOUR, to address these issues.

III. System Model

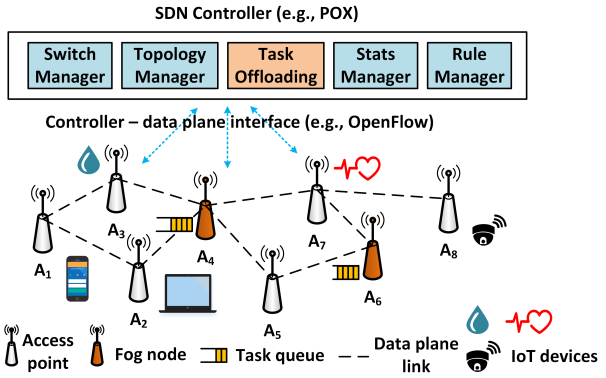


Figure 1: Architecture of SDIoT network

We consider a software-defined fog network as shown in Figure 1, consisting of a set of access points, $\mathcal{A}$, set of fog nodes, $\mathcal{F}$, and set of IoT devices, $\mathcal{D}$. The network is modeled as a directed graph $\mathcal{G} = (\mathcal{A} \cup \mathcal{F}, \mathcal{L})$ where $\mathcal{L}$ denotes the set of links between access points and fog nodes. The access points and the fog nodes are SDN-enabled and can communicate with the SDN controller using southbound APIs [10], [28]. The fog nodes may consist of physical servers or virtual machines (VMs) provisioned on the access points. In general, since provisioning of fog nodes increases the cost of the network, we consider the number of fog nodes to be less than the number of access points, i.e., $|\mathcal{F}| < |\mathcal{A}|$. We consider each device $k \in \mathcal{D}$ has a single computation task at a time, given as t_k. A task is defined as $t_k := (\omega_k, s_k)$, where ω_k and s_k denote the CPU cycles needed for execution and input data size, respectively [23], [24]. Since number of fog nodes is less than the number of access points, task offloading requests from the devices arrive at the fog nodes in a multi-hop manner. The SDN controller makes optimal decisions about where to send each task offloading request and accordingly places appropriate flow-rules in the SDN-enabled access points.

The multi-hop task offloading problem (MHTO) in SDN is formulated as an integer program and is presented below. The key notations are summarized in Table II.

A. Delay Model

We define binary variables $z_k \; \forall k \in \mathcal{D}$ to represent whether task t_k is computed locally ($z_k = 0$) or offloaded to a fog device ($z_k = 1$). The time taken for local execution of task t_k is given as $\delta_k^{\text{loc}} = \frac{\omega_k}{f_k}$, where f_k represents the CPU frequency of device $k \in \mathcal{D}$. On the other hand, if task t_k is offloaded, the time taken comprises of — a) time δ_k^{tx} to transmit the data s_k to the associated access point $i \in \mathcal{A}$, b) propagation delay δ_k^{prp} from access point i to fog node $j \in \mathcal{F}$, c) queuing delay δ_j^{que} at fog node j, and d) task execution time δ_k^{fog} at the fog nodes.

We consider a log-distance path-loss model with log-normal shadowing given as $PL_{[dB]} = 140.7 + 36.7 \log_{10} d_{[km]} + \mathcal{N}(8)$ [21]. Therefore, the maximum data rate between device $k \in \mathcal{D}$ and access point $i \in \mathcal{A}$ is given by Shannon's equation as $r_{ki} = B \log_2(1 + \frac{p_k^{tx} - PL_{ki}}{\sigma^2})$, where p_k^{tx} represents the transmission signal power of device k, and σ^2 represents the noise power. In this work, we mainly focus on the choice of fog node, which is, in general, more than 1-hop away from the devices. Therefore, we consider device $k \in \mathcal{D}$ can access the network by associating with access point $i \in \mathcal{A}$, using existing association policies $\mathcal{X}$, such as [29], such that $\mathcal{X}(k) = i$. Thus, the time taken to transmit data s_k from device k for offloading is given as $\delta_k^{\text{tx}} = \frac{s_k}{r_{k\mathcal{X}(k)}}$. We define binary variables $x_{ij}^k \; \forall (i,j) \in \mathcal{L}$ to denote whether link $(i,j) \in \mathcal{L}$ is chosen for offloading task t_k. Let δ_{ij} be the propagation delay associated with each link $(i,j) \in \mathcal{L}$. Therefore, the propagation delay experienced by an offloaded task t_k, from access point to fog node, is given as $\delta_k^{\text{prp}} = \sum_{ij} \delta_{ij} x_{ij}^k$. At each fog node $j \in \mathcal{F}$, offloaded tasks arrive following unique paths consisting of sequence of links $(i,j) \in \mathcal{L}$, through the network. Therefore, applying the Kleinrock independence approximation [30], the task arrival at a particular fog node $j \in \mathcal{F}$ may be approximated as a Poisson process, and task arrival rate given as $\gamma_j = \sum_{ik} x_{ij}^k$. We consider a multi-threaded model of task execution at the fog nodes. When a task arrives at a fog node, it waits in the task queue while a proper application for processing the task is fetched. This allows parallel processing of multiple tasks. Therefore, considering an M/M/1 model, the queuing delay at the fog node $j \in \mathcal{F}$ is given as $\delta_j^{\text{que}} = \frac{1}{\mu_j - \gamma_j}$, where μ_j denotes the service rate. We consider binary variables $y_j^k \; \forall k \in \mathcal{D}$ to denote whether fog node $j \in \mathcal{F}$ is chosen to offload task t_k. Therefore, the time taken for execution of task t_k at the chosen fog node ($y_j^k = 1$) is given as $\delta_k^{\text{fog}} = \sum_j \frac{\omega_k}{f_j} y_j^k$, where f_j represents the CPU frequency of fog node j. Here, we assume that tasks are unsplittable, i.e., a particular task can be offloaded to only one fog node. Therefore, we define a cost function for delay in the MHTO problem as $J_\delta(x,y,z) := \sum_k \left[(1 - z_k)\delta_k^{\text{loc}} + z_k(\delta_k^{\text{tx}} + \delta_k^{\text{prp}} + \delta_k^{\text{fog}})\right] + \sum_j \delta_j^{\text{que}}$.

B. Energy Consumption Model

The energy consumed during task execution on an IoT device depends on various factors such as task type and clock frequency. In line with existing works [21], [23], [24] we consider the energy consumption of local computation to be given as $\mathcal{E}_k^{\text{loc}} = \rho(f_k)^2 \omega_k$, where ρ is the power coefficient depending on chip architecture and f_k is the CPU frequency of device $k \in \mathcal{D}$. On the other hand, when t_k is offloaded, the energy consumption of an IoT device is given by the energy consumed for transmitting the data s_k. The SDN controller is

Table I: Summary of existing literature

Work	Delay	Energy	Multihop	SDN	Rule-capacity
Chiti *et al.* [17], Yousefpour *et al.* [19]	✓	✗	✗	✗	✗
Chang *et al.* [16], Shah-Mansouri and Wong [18], Zhao *et al.* [25], Tran *et al.* [21]	✓	✓	✗	✗	✗
Huang *et al.* [22], Chen and Hao [24], Zhao *et al.*	✓	✓	✗	✓	✗
Proposed scheme (DETOUR)	✓	✓	✓	✓	✓

Table II: Summary of key notations

Notation	Description
$\mathcal{A}$	Set of SDN-enabled access points.
$\mathcal{F}$	Set of SDN-enabled fog nodes.
$\mathcal{L}$	Set of links between $\mathcal{A} \cup \mathcal{F}$.
$\mathcal{D}$	Set of IoT devices.
t_k	Computation task of device $k \in \mathcal{D}$.
ω_k	CPU cycles needed for execution of task t_k.
s_k	Input data size for task t_k.
$\mathcal{X}$	Access point association policy.
δ_k^{loc}	Time taken for local computation of task t_k.
δ_k^{tx}	Time taken to transmit data s_k to access point.
δ_k^{prp}	Time taken to reach fog node from access point.
δ_j^{que}	Queuing delay at fog node $j \in \mathcal{F}$.
δ_k^{fog}	Task execution time at fog node.
f_k	CPU frequency of device $k \in \mathcal{D}$.
f_j	CPU frequency of fog node $j \in \mathcal{F}$.
μ_j	Task service rate of fog node $j \in \mathcal{F}$.
$\mathcal{E}_k^{\text{loc}}$	Device energy consumption for executing task t_k.
$\mathcal{E}_k^{\text{tx}}$	Device energy consumption for transmitting data s_k.
$\mathcal{B}^{\text{wl}}$	Bandwidth of wireless channel.
$\mathcal{B}_{ij}$	Bandwidth of link $(i,j) \in \mathcal{L}$.
$\mathcal{R}_i^{\max}$	Maximum SDN rule-capacity at access point $i \in \mathcal{A}$.

capable of collecting information such as transmission power and data rate using protocols through southbound APIs such as Simple Network Management Protocol (SNMP) or Control and Provisioning of Wireless Access Points (CAPWAP) [10]. Therefore, the energy required for offloading task t_k is given as $\mathcal{E}_k^{\text{tx}} = p_k^{tx} \delta_k^{\text{tx}}$. Thus, we define a cost function for energy in the MHTO problem as $J_{\mathcal{E}}(x,y,z) \coloneqq \sum_k \left[(1-z_k)\mathcal{E}_k^{\text{loc}} + z_k \mathcal{E}_k^{\text{tx}}\right]$.

C. Problem Formulation

Using the cost functions for delay and energy, we formulate the MHTO problem for SDN as follows:

$$\min_{x,y,z} \quad J(x,y,z) = \alpha J_\delta(x,y,z) + (1-\alpha) J_{\mathcal{E}}(x,y,z) \tag{1a}$$

$$\text{s.t.} \quad \sum_j x_{ij}^k - \sum_j x_{ji}^k = \begin{cases} +1, & \text{if } i = \mathcal{X}(k), \\ -y_i^k, & \text{if } i \in \mathcal{F}, \\ 0, & \text{otherwise.} \end{cases} \tag{1b}$$

$$\sum_{j \in \mathcal{A}} y_j^k = 1, \quad \forall k \in \mathcal{D} \tag{1c}$$

$$(1-z_k)\mathcal{E}_k^{\text{loc}} + z_k \mathcal{E}_k^{\text{tx}} \le \mathcal{E}_k^{\max}, \quad \forall k \in \mathcal{D} \tag{1d}$$

$$\sum_k \frac{1}{\delta_k^{\text{tx}}} s_k x_{ij}^k \le \mathcal{B}_{ij}, \quad \forall (i,j) \in \mathcal{L} \tag{1e}$$

$$\sum_{jk} x_{ij}^k \le \mathcal{R}_i^{\max}, \quad \forall i \in \mathcal{A} \tag{1f}$$

where α is a user-defined constant to control the relative importance of delay and energy consumption. For example, for highly latency critical applications, α can be set to 1. The *flow-conservation constraints* in Equation (1b) ensure that each task is served only by the fog nodes. Equation (1c) ensures that each task can be offloaded to only one fog node, i.e., tasks are unsplittable. The *energy constraints* in Equation (1d) take into account the energy of the IoT devices, so that devices are not depleted of energy. Equation (1e) takes into account the bandwidth utilization of the links, in order to reduce congestion in the network, and hence, reduce the number of retransmissions. Equation (1f) ensures that the number of SDN flow-rules at the access points does not exceed the maximum capacity. In order to simplify the model, we consider exact-match flow-rules [31], where a flow-rule is placed for each task t_k.

The MHTO integer program presented in Equation (1) is difficult to solve since — a) the feasible set is non-convex due to the presence of binary variables x, y, z, and b) the objective function in Equation (1a) is non-linear due to the presence of product terms $z_k \delta_k^{\text{prp}}$ and $z_k \delta_k^{\text{fog}}$. Therefore, we adopt the linearization approach proposed in [32] by — a) replacing the product terms by new continuous variables a_k and b_k, $\forall k$, and b) adding four new linear constraints for each product term, in order to transform the problem into an integer *linear* program (ILP) as given below.

$$\min_{x,y,z} \quad J'(x,y,z) = \alpha J'_\delta(x,y,z) + (1-\alpha) J_{\mathcal{E}}(x,y,z) \tag{2a}$$

$$\text{s.t.} \quad 0 \le a_k \le z_k \sum_{ij} \delta_{ij}, \ \forall k \in \mathcal{D}, \tag{2b}$$

$$\delta_k^{\text{prp}} - (1-z_k) \sum_{ij} \delta_{ij} \le a_k \le \delta_k^{\text{prp}}, \ \forall k \in \mathcal{D}, \tag{2c}$$

$$0 \le b_k \le \frac{\max_k \omega_k}{\min_j f_j}, \ \forall k \in \mathcal{D}, \tag{2d}$$

$$\delta_k^{\text{fog}} - (1-z_k)\frac{\max_k \omega_k}{\min_j f_j} \le b_k \le \delta_k^{\text{fog}}, \ \forall k \in \mathcal{D}, \tag{2e}$$

(1b)–(1f).

where $J'_\delta(x,y,z) = \sum_k \left[(1-z_k)\delta_k^{\text{loc}} + z_k \delta_k^{\text{tx}} + a_k + b_k\right] + \sum_j \delta_j^{\text{que}}$ represents the linearized form of $J_\delta(x,y,z)$, and Equations (2b)–(2e) represent the additional constraints introduced due to linearization.

Moderate-sized instances of the ILP in Equation (2) can be solved exactly in reasonable time, using commercial solvers such as CPLEX or Gurobi [33]. However, with large instances, the ILP may take prohibitively high time to converge, which may not be acceptable for the on-line nature of task offloading under consideration. Therefore, in the subsequent section, we present an approximate greedy solution to the problem. Further, to validate the proposed greedy algorithm, we compare

its performance against the ILP solution.

D. Approximate Greedy Solution

In order to design a greedy heuristic solution to the MHTO problem, we define utility functions as follows:

$$\mathcal{U}_k^{\text{off}} := \beta_\delta \frac{\delta_k^{\text{loc}} - \delta_k^{\text{tx}}}{\delta_k^{\text{loc}}} + \beta_\mathcal{E} \frac{\mathcal{E}_k^{\text{loc}} - \mathcal{E}_k^{\text{tx}}}{\mathcal{E}_k^{\text{loc}}} \tag{3a}$$

$$\mathcal{U}_{jk}^{\text{fog}} := \theta_{\text{que}} \delta_j^{\text{que}} + \theta_{\text{exe}} \frac{\omega_k}{f_j} \frac{\min_j f_j}{\max_k \omega_k} \tag{3b}$$

$$\mathcal{U}_{ij}^{\text{link}} := \left(\gamma_\delta \frac{\delta_{ij}}{\max_{ij} \delta_{ij}} + \gamma_\mathcal{B} \frac{\mathcal{B}_{ij}^{\text{util}}}{\mathcal{B}_{ij}} + \gamma_\mathcal{R} \frac{\mathcal{R}_i^{\text{util}}}{\mathcal{R}_i^{\max}} \right)^{-1} \tag{3c}$$

Equation (3a) represents the offload utility which incorporates the improvement in delay and energy consumption by offloading a task. The constants β_δ and $\beta_\mathcal{E}$ denote the relative importance of delay and energy consumption, respectively, which are application dependent. It is noteworthy that a task t_k should be offloaded only when $\mathcal{U}_k^{\text{off}} \geq 0$. Similarly, given that a particular task t_k is chosen to be offloaded, Equation (3b) denotes the utility of choosing a particular fog node $j \in \mathcal{F}$, which takes into account the queuing delay, as well as the task execution time. The constants θ_{que} and θ_{exe} denote the relative importance of queuing delay and task execution time, respectively. For example, if the service rate of the fog nodes is low, we should set $\theta_{\text{que}} > \theta_{\text{exe}}$ to reduce the overall delay. In our experiments, we assume $\theta_{\text{que}} = \theta_{\text{exe}}$. Equation (3c) denotes the utility of choosing link $(i, j) \in \mathcal{L}$. The constants γ_δ, $\gamma_\mathcal{B}$, and $\gamma_\mathcal{R}$ denote the relative importance of link delay, bandwidth utilization, and SDN rule utilization respectively, which are application dependent. For example, if the memory available at the SDN switches is low, $\gamma_\mathcal{R}$ should be relatively high compared to γ_δ and $\gamma_\mathcal{B}$. It is noteworthy that link delay, δ_{ij}, link bandwidth utilization, $\mathcal{B}_{ij}^{\text{util}}$, and rule utilization, $\mathcal{R}_i^{\text{util}}$, are available in real-time at the SDN controller using tools such as OpenNetMon [34], and does not require *apriori* knowledge of all flows. Therefore, the utilities defined in Equation (3) can be calculated online, upon task arrival.

Algorithm 1 Detour: Multi Hop Task Offloading

Inputs: $\mathcal{A}, \mathcal{F}, \mathcal{L}, \mathcal{D}, \mathcal{X}$
Output: Task offloading policy $\langle z_k, y_k, p_k \rangle \; \forall t_k \mid k \in \mathcal{D}$.

1: **for** task $t_k \mid k \in \mathcal{D}$ **do**
2: Calculate $\mathcal{U}_k^{\text{off}}$ using Equation (3a).
3: **if** $\mathcal{U}_k^{\text{off}} \geq 0$ **then**
4: Set $z_k \leftarrow 1$.
5: **for** fog node $j \in \mathcal{F}$ **do**
6: Calculate $\mathcal{U}_{jk}^{\text{fog}}$ using Equation (3b).
7: Select fog node $y_k = \text{argmin}_j \mathcal{U}_{jk}^{\text{fog}}$.
8: Calculate offload path p_k from access point $\mathcal{X}(k)$ to fog node y_k using Equation (3c) and Dijkstra's shortest path algorithm.
9: **else**
10: Set $z_k \leftarrow 0$, compute t_k locally, at device k.

Algorithm 1 presents the proposed greedy algorithm, termed DETOUR, for the MHTO problem in SDN. Upon arrival of a task t_k, we decide whether to offload it or not, according to local information available at that instant, as given in Step 3. If the task is chosen for offloading, the best fog node for that task is chosen using the fog utility as given in Step 7. Subsequently, the offload path from the associated access point (given by association policy $\mathcal{X}$) to the selected fog node is decided while taking into account the dynamic network conditions such as link and rule utilization, as given in Step 8. We analyze the complexity of the proposed scheme by considering the time complexity of Algorithm 1. For a given task, t_k, Step 3 takes constant time. Step 7 involves finding the minimum in a list of length $|\mathcal{F}|$ and has worst-case time $\mathcal{O}(|\mathcal{F}|)$. Subsequently, Dijkstra's algorithm in Step 8 has worst-case running time of $\mathcal{O}((|\mathcal{A}|+|\mathcal{F}|)^2)$. Therefore, the worst-case time complexity is given as $\mathcal{O}(|\mathcal{F}| + (|\mathcal{A}| + |\mathcal{F}|)^2) \approx \mathcal{O}((|\mathcal{A}| + |\mathcal{F}|)^2)$.

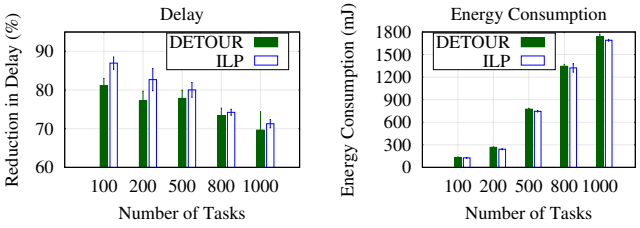


Figure 2: Comparison of DETOUR vs ILP-based solution

We used the commercial Gurobi solver [33] to solve the ILP formulated in Equation (2). Figure 2 shows the comparison between the proposed scheme, DETOUR, and the ILP-based solution. We observe that with a large number of tasks, the proposed scheme offers comparable performance to the ILP, and thus, offers an approximate solution to the MHTO problem in SDN.

IV. Performance Evaluation

A. Simulation Settings

We evaluate the performance of the proposed scheme using the POX[2] SDN controller and the Mininet[3] network emulator. The experiments were carried out on a Intel i7 2.7 GHz PC with 8 GB RAM, running Linux kernel 4.15. The different parameters considered for the experiments are given in Table III. In line with existing literature [19], we consider Arduino and Intel i7 CPU as examples of IoT device and fog node, respectively, in order to obtain realistic values regarding computation capability. Since topology traces of software-defined access networks are difficult to obtain, in this work, we consider a scale-free Barabasi-Albert topology [35]. The access points and fog nodes are arranged in a 500m x 500m area, as shown in Figure 3. The figure shows four different topologies considered in this work, with different selection of fog nodes. The fog nodes are randomly chosen from the total number of nodes. The IoT devices are placed uniformly within the coverage area.

[2]https://github.com/noxrepo/pox
[3]http://mininet.org/

Table III: Simulation parameters

Parameter	Value
Number of APs	15
Number of fog nodes	5
Number of tasks	100 – 1000
Fog CPU frequency	2.9 – 4.2 GHz
IoT device CPU frequency	16 – 84 MHz
IoT device transmit power	60 mW [19]
Computation amount for task	1500 – 2500 Megacycles [21]
Battery capacity of IoT device	1000 J [24]
Average task size	450KB [21]
Noise power	−100 dB [21]
Wireless channel bandwidth	20 MHz [21]

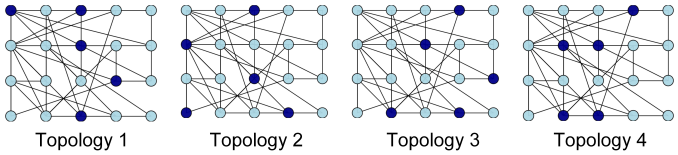


Figure 3: Topologies considered for experiment with different selection of fog nodes (fog nodes are given in blue)

B. Benchmark Schemes

To show the effectiveness of the proposed scheme, DETOUR, we compare it with the following baselines — delay-aware greedy-path (DAGP) [24], and random-offloading random-path (RORP). In the DAGP scheme, the offload decision is taken in order to minimize the average delay while respecting the energy constraints. The shortest path to the fog node is taken based on the hop count. The DAGP scheme was chosen to show the impact of factors other than delay and energy, such as — multi-hop path, link utilization, and SDN rule-capacity, on software-defined task offloading schemes. In the ROGP scheme, the offload decision and path to the fog node are randomly chosen. Since the RORP scheme takes path selection decisions randomly, it highlights the impact of proper path selection in a multi-hop software-defined task offloading scenario. In both the DAGP and RORP schemes, the nearest fog node (minimum number of hops) is chosen as the offload target.

On the other hand, in the proposed scheme, DETOUR, the offload decision, choice of fog node, and path to fog node are taken while taking into account — delay, energy, link utilization and SDN rule-capacity, according to Algorithm 1.

C. Results and Discussion

1) Average Reduction in Delay: We analyze the average reduction in delay per task due to task offloading. Figure 4 shows the performance of the proposed scheme DETOUR compared to the benchmarks. From the figure, we observe that with an increase in the number of tasks, the reduction in delay decreases. This is expected, since more offloaded tasks increases the load on the network (in terms of bandwidth utilization) and load on the fog nodes (queuing delay). We also observe that the performance varies according to the topology (i.e., different selection of fog nodes). However, the proposed scheme, DETOUR, outperforms the benchmarks in all cases. In particular, overall, the proposed scheme is capable of reducing the delay by 35% and 12% more, compared to the RORP and DAGP schemes, respectively. The RORP scheme does not take into account the path chosen during offloading, and hence suffers from increased delay due to longer path lengths. On the other hand, the DAGP scheme chooses the shortest path based on hop count and does not take into account the bandwidth utilization and SDN rule capacity. Thus it suffers additional delay due to congestion and re-transmission.

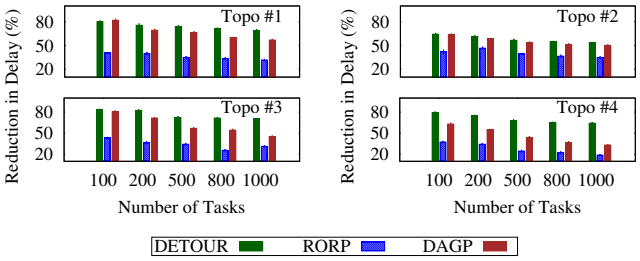


Figure 4: Percentage reduction in delay vs tasks

2) Average Energy Consumption: We also analyze the average energy consumption per device due to task offloading. Figure 5 shows the performance of the proposed scheme compared to the benchmarks. From the figure, we observe that with an increase in the number of tasks, the energy consumption increases almost linearly. We also observe that the performance varies according to the topology. However, the proposed scheme, DETOUR, outperforms the benchmarks in all cases. In particular, overall, the proposed scheme is capable of reducing the energy consumption by 13% and 21%, compared to the RORP and DAGP schemes, respectively. The RORP scheme randomly chooses whether to offload a task, and thus, suffers increased energy consumption due to local processing. On the other hand, the DAGP chooses to offload tasks only based on delay without considering whether offloading that particular task will reduce energy consumption. Therefore, DAGP suffers from increased energy consumption due to both sub-optimal offloading decision, as well as re-transmissions (as explained in Section IV-C1).

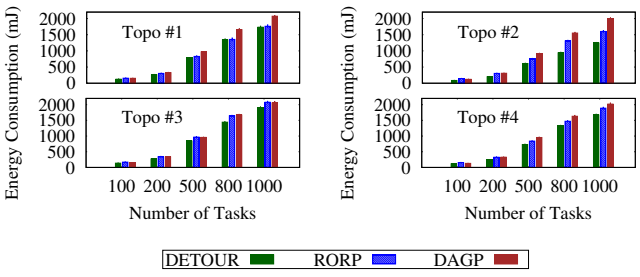


Figure 5: Energy consumption vs tasks

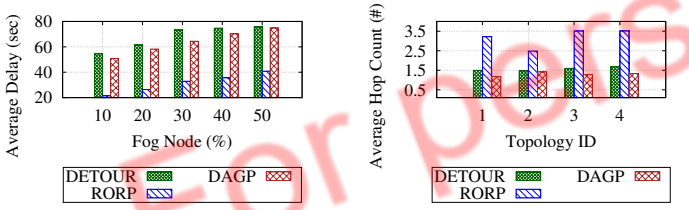
3) Impact of Intermediate Hop Count: We analyze the impact on intermediate hop count to the fog nodes on task offloading, as shown in Figure 6. Table IV shows the average number of hops to the nearest fog node, with different percentages of nodes acting as fog nodes. It is evident that with increasing number of fog nodes, the average hop count to the nearest fog node decreases.

Table IV: Fog nodes vs hop count

Fog nodes (%)	10	20	30	40	50
Hop Count	2.166	1.5	1.21	1.08	1.0

Figure 6(a) shows the percentage reduction in delay (due to offloading) with different percentages of fog nodes (See Table IV), while the number of tasks is kept constant at 500. We observe that while the proposed scheme, DETOUR outperforms the benchmarks, the relative performance is better with less number of fog devices. In particular, with 30% of the APs acting as fog devices, DETOUR is capable of achieving 40% and 9% more reduction in delay compared to the RORP and DAGP schemes, respectively. This implies that the proposed scheme is capable of achieving good performance with less number of fog devices, thereby reducing CAPEX and OPEX.

Figure 6(b) shows the average hop count incurred by the different schemes, across the different topologies, while task offloading. From the figure, we observe that the RORP scheme has the highest hop count, while the DAGP scheme has the lowest. It is interesting to note that even with the lowest hop count, the DAGP scheme does not have the best performance in terms of delay (refer Figure 6(a)). This implies that the delay performance depends not only on hop count, but other factors such as bandwidth utilization and SDN rule-capacity, which are taken into account by the proposed scheme.

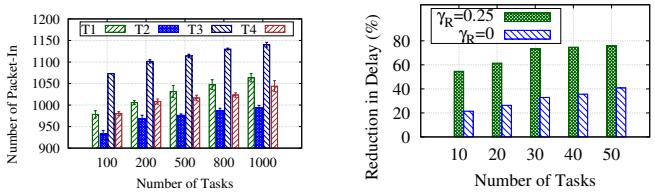


(a) Delay reduction vs % fog nodes (b) Avg. hop count with 500 tasks

Figure 6: Impact of intermediate hop count

4) Impact of SDN on Task Offloading: Even though SDN offers advantages in task offloading in terms of gathering network statistics, and reducing delay and energy consumption (Sections IV-C1 – IV-C3), it comes at a price, in the form of two factors — control plane overhead, and b) additional flow-rule utilization. In this work, we have utilized the OpenNetMon [34] framework to collect information such as delay, link utilization and rule utilization. OpenNetMon uses adaptive polling techniques and probe packets for measurement, which introduce additional control plane overhead in the network. Further, in order to forward probe packets in a timely manner, OpenNetMon uses fine-grained exact-match rules, which increase rule utilization in the network. Since the rule-capacity of SDN switches is limited [36], this adversely effects the performance of task offloading.

In this Section, we analyze the impact of SDN on task offloading, as shown in Figure 7. We measure the number of packet-in messages to the controller to get an estimate of the SDN control plane overhead. Packet-in messages are generated when a flow[4] does not find a matching flow-rule at the SDN switch. Figure 7(a) shows the number of packet-in messages with increasing number of tasks. We observe that with an increase in the number of tasks, the growth rate of packet-in messages reduces. This is due to the fact that with increasing number of tasks, the probability of finding a matching flow-rule at the SDN switches increases. Accordingly, less number of packet-in messages are generated for new flows. This implies that the SDN control overhead does not linearly scale with the number of tasks, and is relatively more with less number of tasks.



(a) No. of packet-in vs tasks

Figure 7: Impact of SDN on task offloading

Figure 7(b) shows the impact of SDN rule-capacity on task offloading. The figure shows the average reduction in delay with increasing number of tasks at different values of $\gamma_{\mathcal{R}}$ in Equation (3c). From the figure, we observe that with less number of tasks, $\gamma_{\mathcal{R}} = 0$ (i.e., rule utilization not considered) performs as well as $\gamma_{\mathcal{R}} = 0.25$ (rule utilization considered). However, with increasing number of tasks, considering the effect of rule utilization gives significantly better results. This is because when the rule-capacity of an SDN switch is fully utilized, an existing rule needs to be replaced, which incurs an additional flow-setup delay of $3 - 8$ ms per packet [37], leading to increased overall delay.

From the analysis above, we see that the proposed scheme, DETOUR, is able to reduce the average delay and energy consumption in the network, by utilizing the global network view offered by the SDN controller, and taking into consideration dynamic network conditions such as link and rule utilization.

V. Conclusion

In this paper, we proposed a task offloading scheme for software-defined networks where IoT devices are connected to fog computing nodes by multi-hop IoT access points. The global view of the network at the SDN controller was used to take optimal decisions about task offloading, while considering dynamic network conditions. Since the non-linearity of the task offloading problem makes it hard to solve, we utilized a linearization technique to present an integer linear programming (ILP) formulation of the problem. Subsequently, we proposed a greedy-heuristic based scheme to solve the problem efficiently. Experimental results showed that the proposed scheme is capable of reducing the average delay and energy consumption by 12% and 21%, respectively, compared to the state-of-the-art.

[4]A stream of packets representing an end-to-end connection.

In this work, a static topology was considered, i.e., the access points and the fog nodes were considered fixed. However, in a realistic IoT scenario, mobile access points may be present. Therefore, we plan to consider how a dynamic topology will impact the performance of SDN-based task offloading scheme, as a future extension of this work.

REFERENCES

[1] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, 2017.

[2] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[3] K. Dolui and S. K. Datta, "Comparison of Edge Computing Implementations: Fog Computing, Cloudlet and Mobile Edge Computing," in *Proc. of the Global Internet of Things Summit*, June 2017, pp. 1–6.

[4] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things J.*, vol. 3, no. 6, pp. 854–864, 2016.

[5] "OpenFog Reference Architecture for Fog Computing," OpenFog Consortium, Tech. Rep., 2017. [Online]. Available: https://www.openfogconsortium.org

[6] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN)," Protocol Specification Version 1.2, 2013.

[7] Z. Shelby, K. Hartke, and C. Bormann, " The Constrained Application Protocol (CoAP)," Internet Requests for Comments, RFC Editor, RFC 7252, 2014.

[8] K. Sood, S. Yu, and Y. Xiang, "Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A Review," *IEEE Internet of Things J.*, vol. 3, no. 4, pp. 453–463, 2016.

[9] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things J.*, vol. 4, no. 6, pp. 1994–2008, 2017.

[10] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "OpenRoads: Empowering Research in Mobile Networks," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 125–126, 2010.

[11] A. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Towards a Software-defined Network Operating System for the IoT," in *Proc. of the IEEE World Forum on Internet of Things*, Dec 2015, pp. 579–584.

[12] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled Software Sefined Networking for Efficient and Scalable IoT Communications," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 48–54, 2015.

[13] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic, "Software-defined fog network architecture for iot," *Springer Wireless Personal Communications*, vol. 92, no. 1, pp. 181–196, 2017.

[14] H. Gupta, S. B. Nath, S. Chakraborty, and S. K. Ghosh. (2016) SDFog: A Software Defined Computing Architecture for QoS Aware Service Orchestration over Edge Devices. [Online]. Available: arXiv:1609.01190

[15] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing Wireless Fog Networks using Software-Defined Networking," in *Proc. IEEE/ACS Int. Conf. Computer Systems and Applications*, 2017, pp. 1149–1156.

[16] Z. Chang, Z. Zhou, T. Ristaniemi, and Z. Niu, "Energy Efficient Optimization for Computation Offloading in Fog Computing System," in *Proc. of the IEEE GLOBECOM*, Dec 2017, pp. 1–6.

[17] F. Chiti, R. Fantacci, and B. Picano, "A Matching Theory Framework for Tasks Offloading in Fog Computing for IoT Systems," *IEEE Internet of Things Journal*, pp. 1–1, 2018, doi: 10.1109/JIOT.2018.2871251.

[18] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018.

[19] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, April 2018.

[20] L. Huang, X. Feng, L. Qian, and Y. Wu, "Deep Reinforcement Learning-Based Task Offloading and Resource Allocation for Mobile Edge Computing," in *EAI Int. Conf. on Machine Learning and Intelligent Communications*, 2018, pp. 33–42.

[21] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, Jan 2019.

[22] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency MEC framework for SDN-based LTE/LTE-A networks," in *Proc. of the IEEE ICC*, 2017, pp. 1–6.

[23] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, and Y. Zhang, "Software Defined Cooperative Offloading for Mobile Cloudlets," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1746–1760, 2017.

[24] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[25] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal Placement of Cloudlets for Access Delay Minimization in SDN-Based Internet of Things Networks," *IEEE Internet of Things J.*, vol. 5, no. 2, pp. 1334–1344, April 2018.

[26] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[27] C. Huang, M. Chiang, D. Dao, W. Su, S. Xu, and H. Zhou, "V2V Data Offloading for Cellular Network Based on the Software Defined Network (SDN) Inside Mobile Edge Computing (MEC) Architecture," *IEEE Access*, vol. 6, pp. 17 741–17 755, 2018.

[28] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards Programmable Enterprise WLANS with Odin," in *Proc. of the ACM HotSDN*, 2012, pp. 115–120.

[29] S. Bera, S. Misra, and M. S. Obaidat, "Mobi-Flow: Mobility-Aware Adaptive Flow-Rule Placement in Software-Defined Access Network," *IEEE Transactions on Mobile Computing*, p. 1, 2018.

[30] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.

[31] "Openflow version 1.5.0," https://www.opennetworking.org/, accessed: 2017-06-24.

[32] F. Glover, "Improved Linear Integer Programming Formulations of Nonlinear Integer Problems," *Management Science*, vol. 22, no. 4, pp. 455–460, 1975.

[33] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com

[34] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *Proc. IEEE NOMS*, 2014, pp. 1–8.

[35] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[36] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," in *Proc. IEEE INFOCOM*, 2013, pp. 545–549.

[37] K. He, J. Khalid, A. Gember-Jacobson, S. Das, C. Prakash, A. Akella, L. E. Li, and M. Thottan, "Measuring Control Plane Latency in SDN-enabled Switches," in *Proc. of the ACM SIGCOMM SOSR*, 2015, pp. 25:1–25:6.